Array in Java

1. What is array?

An array is a contiguous group of memory locations that can store a set of homogenous data elements. An array of size n is a group of n number of cells in the memory that can store n number values.

To work with arrays in a program, we need the following three steps-

1. Declaration of the array
2. Initialization of the array
3. Retrieval of the array elements

2. Types of arrays

Arrays can be broadly classified into two categories –

2.1 One-dimensional array

In a one-dimensional (in short 1-D) array, memory allocation takes place for n number of cells, where n is the size of the array. Also called a single-dimensional array.

2.2 Multi-dimensional array

Multi-dimensional arrays can be further categorized into two-dimensional arrays, three-dimensional arrays etc. It can be up to n-dimension depending upon the availability of the memory. But logically, an array up to three dimensions can be visualized and handled easily.

- Two-dimensional (2-D) array is represented by rows and columns and the total size of the array is number of rows x number of columns. A two-dimensional array can be visualized as a table of m rows and n columns. It can also be viewed as multiple 1-D arrays.
- Three-dimensional array is represented by rows, columns and depth and the total size of the array is rows x columns x depth. A three-dimensional array can be visualized as a cube of m rows x n columns x p depth.

3. Memory representation of arrays

3.1 Declaration of an array in Java

In Java declaration of the array is done through the **new** keyword. The syntax of array declaration is as follows:

<data type> array_name[ ][ ][ ] = new <data type>[size][size][size];

Here [ ] represents dimensions in the array

When an array is declared, memory allocation takes place according to the dimension and size. Each cell of the array is referred by one index or subscript value. The index starts from 0.

Few examples –

1. To create an integer array (1-dimension) of size 4:

```
int array[ ] = new int [4];
   or
int [ ]array = new int [4];
```

0	1	2	3	
1st	2nd	3rd	4th	Cells

2. To create an integer array (2-dimension) of 4 rows and 3 columns:

```
int array[ ][ ] = new int [4][3]; // 4 rows and 3 columns
        or
int [ ][ ]array = new int [4][3];
```

	0	1	2
0			
1			
2			
3			

3. To create an integer array (3-dimension) of 4 rows, 3 columns and depth 2:

```
int array[ ][ ][ ] = new int [2][4][3];
        or
int [ ][ ][ ]array = new int [2][4][3];
```

	0	1	2
0			
1			
2			
3			

[0]

	0	1	2
0			
1			
2			
3			

[1]

3.2 Memory representation of the array

1-D array is represented in the memory as a sequential block of memory cells as shown below:

0	1	2	3

2-D array is represented in two ways in the memory –
Sample 2-D array [4x3] i.e., 4 rows and 3 columns
int arr[][] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

i. Row Major order: Rows get priority over columns

0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2	3,0	3,1	3,2
1	2	3	4	5	6	7	8	9	10	11	12

ii. Column Major order: Columns get priority over rows

0,0	1,0	2,0	3,0	0,1	1,1	2,1	3,1	0,2	1,2	2,2	3,2
1	4	7	10	2	5	8	11	3	6	9	12

3.3 Address Calculation in Array (For ISC Students)

In a 1-dimensional Array, the address of the required element says A[i] can be calculated by using the formula given below:

ADDRESS OF A[I] = B + W*(I – I_0) , where

B is the Base Address
W is the size of each array cell (according to the size of the data type of the array)
I is the index position of the required array cell
I_0 is the index position of the cell whose address is given as the base address.

In the case of a 2-dimensional Array, the address of the required element say A[i][j] can be calculated as per the memory arrangement of the array – (row major or column major) order. Hence there are two formulas for the address calculation.

For Row Major order, the formula is -

ADDRESS OF A[I][J] = B + W*[(I – I_0)*C + (J – J_0)] , where

B is the Base Address
W is the size of each array cell (according to the size of the data type of the array)
I, J is the index position of the required array cell
I_0, J_0 is the index position of the cell whose address is given as base address.
C is the total number of columns in the array A.

For Column Major order, the formula is -

ADDRESS OF A[I][J] = B + W*[(I – I_0) + (J – J_0)*R] , where

B is the Base Address
W is the size of each array cell (according to the size of the data type of the array)
I, J is the index position of the required array cell
I_0, J_0 is the index position of the cell whose address is given as base address.
R is the total number of rows in the array A.

Questions on Address finding:

An array ARR[35][15] is stored in the memory along the row with each of its element occupying 4 bytes. Find out the base address and the address of an element ARR[20][5], if the location of ARR[0][0] i.e base address is 2872, ARR[2][2] is stored at the address 3000. The array is in row-major order.

Solution:

Formula for the address in Row major order arrangement = B + W x (Cx(i-i_0)+(j-j_0))

B = A - Wx(Cx(i-i_0)+(j-j_0))
= 3000 – 4*(15*(2-0)+(2-0)) = 3000 – 4*(30+2) = 3000 – 128 = 2872

A = 2872 + 4*(15*(20-0)+(5-0))
= 2872 + 4*(300+5) = 2872 + (4*305) = 2872 + 1220 = 4092

Each element of an array A[-15…30][20…25] requires 4 bytes for storage. Find the address of A[30][20] is the base address for the cell A[-10][-10] is 1000 and the array is in column-major order. Show the complete work.

Solution:

Address[i][j] = B + W*((i-i_0)+R(j-j_0)); R = 30-(-15)+1=46

= 1000 + 4*(30-(-10)+46*(20-(-10))

= 1000 + 4*(30+46*30)

= 1000 + 4*1410 = 6640

3.4 Initialization of the array

An array must be initialized with a valid set of values after its declaration. An array can be initialized in two ways.

- **Static initialization of 1D array**

With static initialization, the array will get a fixed set of values during the execution of the program. Actually, in this case, the programmer himself or herself provides the values to the array within the program.

Given below some examples with 1-D array:

i. An array will hold the following set of integers: 10, 0, 23, 26, 0.

```
int arr[ ]=new int[5];
arr = {10,0,23,26,0};
```

0	1	2	3	4
10	0	23	26	0

ii. An array of character type is holding all the five vowels : a, e, I, o ,u.

```
char alpha[ ]=new char[5];
alpha={'a', 'e', 'I', 'o', 'u'};
```

0	1	2	3	4
a	e	I	o	u

iii. An array of String type to hold any five names

```
String names[ ]=new String[5];
names={"Akash", "Bikash", "Lokesh", "Prakash", "Sukesh"};
```

0	1	2	3	4
Akask	Bikash	Lokesh	Prakash	Sukesh

Similarly for 2-D and 3-D array static initialization can be done likewise.

- **Static initialization of 2D array**
  - An array will hold the following set of integers: 1,2,3,4,5,6,7,8,9.

```
int arr[ ][ ]=new int[3][3];
arr = {1,2,3,4,5,6,7,8,9};
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

  - An array of character type is holding all the five alphabets : R, A, I, N, B, O.

```
char alpha[ ][ ]=new char[2][3];
alpha={ 'R', 'A', 'I', 'N', 'B', 'O' };
```

	0	1	2
0	R	A	I
1	N	B	O

  - An array of String type to hold names of six students

```
String names[ ][ ]=new String[3][2];
names={"Akash", "Bikash", "Lokesh", "Prakash", "Sukesh", "Arun"};
```

	0	1
0	Akash	Bikash
1	Lokesh	Prakash
2	Sukesh	Arun

- **Dynamic initialization of the array**

With dynamic initialization, the array will get a variable set of values during the execution of the program. Actually, in this case, the programmer declares the array within the program and during the program execution the array will receives the values from the user itself.

For dynamic initialization of any array, first we need to declare the array within the program and then a separate loop is needed to receive the values during run time.

Example on 1D array:

Given below an example to store names of 10 students and their average marks from the user in a String and a double type array:

```
String names[ ]=new String[10]; //array declaration and allocation of memory
double marks[ ]=new double[10]; //same
for(int i=0; i<10; i++) //loop to accept data and store them to the array
{
        System.out.println("Enter the name and the average marks");
        names[ i ]=sc.nextLine(); //storing the names in the array at I^th index
        marks[ i ]=sc.nextDouble(); //storing the marks in the array at I^th index
}
```

In the above code, sc is an object of Scanner class whose function nextLine() and nextDouble() are used to receive the values from the user during the execution of the program.

Example on 2D array:

Given below an example to store 20 numbers in a 2D array of size 4(row)x5(column) and then compute the average and display them.

```
int num[ ][ ]=new int[4][5]; //4 rows and 5 columns
int avg=0;
for(int i=0; i<4; i++) //loop for rows
{
        for((int j=0; j<5; j++) //loop for columns
        {
                System.out.println("Enter the no.");
                num[ i ][ j ]=sc.nextInt();
                avg+=num[i][j];
        }
}
for(int i=0; i<4; i++)
{
```

```
                for((int j=0; j<5; j++)
                {
                        System.out.print(num[i][j]+"\t");
                }
                System.out.println();//new line
        }
        System.out.println("Average:"+(avg/20));
```

3.5 Determining the array size

To determine the size of an array, we have to use length property. For example,

```
int arr[] = new int[10];
int len= arr.length;
System.out.println("Size of the array is:"+len); //it is 10 as answer.
```

Note – Kindly notice, **length()** is a string function where whereas **length** is a property of an array.

The size of an array in bytes will be the total length * size of the data type.
Here, in this case, it will be 10*4 = 40 bytes. (Since int occupies 4 bytes)

3.6 Array as parameter

An array can be passed as a parameter to a function. In that case, it will be the 'pass by reference' mechanism occurs. So, any change in the array made within the function will be actually occurring in the array passed to the function.

The given program code will demonstrate this through sorting mechanism on an array passed to a function.

```
class SortArray
{
        int arr[]=new int [5];
        void assign()
        {
                arr={10, 12, 5, 11, 7};
        }
        void sort(int a[])
        {
                int p=a.length;
                for(int i=0;i<p-1;i++)
                {       for(int j=0;j<p-i-1;j++)
                        {       if(arr[j]>arr[j+1])
                                {       int t=arr[j];
                                        arr[j]=arr[j+1];
                                        arr[j+1]=t;
                                }
                        }
                }
        }
}
```

3.7 Copying arrays

Solved questions on Array

1. Mention the advantages and disadvantages of the Array?

   Ans: **Advantages:**
   - Large number of element can be accessed by specifying the array name with the index number.
   - Processing of array is simple using Loops.

   Disadvantages:
   - Arrays are always fixed length i.e. the length of an array cannot be decreased or increased
   - An array can hold only one type of Data

2. Write the Java Statement for the following:
   a. A declare statement to store 20 integer in the variable "Salary"
   b. A declare statement to store 40 names

   Ans: a. int [] salary = new int [20];
   b. string [] names = new string [40];

3. Write the output of the following program

```
double db [] = {3.17,2.97,5.97,2.86,9.23};
for( int i=0;i<=4;i++)
{       db[i] +=3;      }
```

   Ans: 6.17, 5.97, 8.97, 5.86,12.23

4. What is the output of the following code statement ?

```
int [] even = { 2,4,6,8,10};
System.out.println (even [0] + " " + even [3]);
```

   Ans: 2 , 8

5. Define different types of array with an example?

   Ans: Java support 2 types of arrays one dimensional array or linear array and two dimensional array.
   Example for a linear array : int arrMonth [] = new new int [12];
   An array arrMonth is created to store 12 values
   Example of a two dimensional array : int marks [] [] = new int [3][2];
   An integer array of marks is created with 4 rows and 3 columns

6. State the different search methods used?

   Ans: Linear search and binary search are different search methods.

7. Mentioned the different between bubble sort and sequential sort while arranging the items in ascending order?

Ans: In bubble sort the adjacent two numbers are compared and the values are swapped. The higher value is placed in the higher index value of the array. The processed repeated until the values are arranged in ascending order.
While in sequential sort, the lowest element from the array is selected and placed in the first index of the array. Similarly, this process is repeated until all the values are store.

8. Difference between sorting and searching?
   Ans: In sorting: The elements of the arrays are arranged in an order. Integers are arranged in either ascending or descending order. Strings are arranged in alphabetic order.
   Searching is a process where a particular element in an array is located and returns the position of the element of the array.

9. Look at the following statement and answer the questions :
   String str [] = { " Delhi", "Kolkata", "Mumbai", "Chennai"};

   a. What is the length of the array?
   b. What data stored in str[2]?
   c. Write the ranges of indices for the array?

   Ans: a. 4 b. Mumbai c. 0 to 3

10. Consider the array with the following data. What would be the order while doing a bubble sort in ascending order during the third iteration?
    77, 45, 95, 10, 89
    Ans: 10, 45, 77, 89, 95
    Working :
    Pass1 45, 77, 10, 89, 95
    Pass2 45, 10, 77, 89, 95
    Pass3 10, 45, 77, 89, 95

11. Consider the array with the following data. What would be the order while doing a selection sort in ascending order during the third iteration?
    77, 45, 95, 10, 89
    Ans: 10, 45, 77, 95, 89
    Working :
    Pass1 10, 45, 95, 77, 89
    Pass2 10, 45, 95, 77, 89
    Pass3 10, 45, 77, 95, 89

Short answer questions:

1. What is the significance of array in programming?
2. Explain the memory allocation of two dimensional arrays in Java programming.
3. What are multi-dimensional arrays? Explain in brief.
4. What are Strings in Java? Write down the use of String class in Java.
5. What is the use of StringBuffer class? Write down the advantages and disadvantages of using StringBuffer in Java Programming.
6. What is sorting? Explain the three sorting techniques in detail:

a. Selection sort
b. Bubble sort
c. Insertion sort

7. What is searching? Explain the following searching techniques:
    a. Linear search in unsorted array
    b. Linear search in sorted array
    c. Binary search in an array sorted in ascending order
    d. Binary search in an array sorted in descending order
8. Solve the followings:
    a. An array m[-2…..5,-1….4] is stored using column major implementation the address of m[0][0] is 176 and the address of m[4][3] is 230. Find the address of m[3][2].
    b. An array A[7,6] is stored in the memory with each element requiring one byte of memory. If the memory address of A[3,4] is 5000, determine the location of 1st element when the array is stored in row major order.
    c. An array A[-6…..5, 10…..14] is stored in the memory with each element requiring one byte of memory. If the memory address of A[3,11] is 5000, determine the location of 1st element when the array is stored in column major order
    d. An array m[-2…..5,-1….4] is stored using column major implementation the address of m[0][0] is 174 and the address of m[4][3] is 230. Find the address of m[3][2].
    e. An array A[7,6] is stored in the memory with each element requiring one byte of memory. If the memory address of A[3,4] is 5000, determine the location of 1st element when the array is stored in row major order.
    f. An array A[-6…..5, 10…..14] is stored in the memory with each element requiring one byte of memory. If the memory address of A[3,11] is 5000, determine the location of 1st element when the array is stored in column major order